Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __

Test Date: **3/25/2024**

Test Case ID#: Party_Constructor_Test

Name(s) of Testers: Ashwin Wariar

Test Description:
Test for class Party to test that the Constructor correctly instantiates with a name associated with it.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getName()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: None

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party objects for testing	Party green(“Green”)	Does not throw an error	Does not throw an error	
3	Test that the Party constructor has the right name	Party green(“Green”)	green.getName() returns “Green”	green.getName() returns “Green”	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_addCandidate_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the addCandidate() function correctly adds a candidate to the vector of candidates, and returns the correct index.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::addCandidate()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Candidate class works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party objects for testing	Party green("Green") Candidate ashwin("Ashwin") Candidate rob("Rob")	Does not throw an error	Does not throw an error	
3	Test that adding ashwin to the Party works and returns correct index		Does not throw an error; green.addCandidate(&ashwin) returns 0	Does not throw an error; green.addCandidate(&ashwin) returns 0	
4	Test that adding rob to the Party works and returns correct index		Does not throw an error; green.addCandidate(&rob) returns 1	Does not throw an error; green.addCandidate(&rob) returns 1	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __

Test Date: 3/25/2024

Test Case ID#: Party_getCandidates_Test

Name(s) of Testers: Ashwin Wariar

Test Description:
Test for class Party to test that the getCandidates() function correctly returns the vector of candidates.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getCandidates()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate class works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**

2	Set up necessary Party objects for testing as well as test vector to compare	Party green(“Green”)
		Candidate ashwin;
		Candidate ashwin2;
		Candidate ashwin3;
		Candidate ashwin4;
		Candidate ashwin5;
		std::vector<Candidate*> test;
			Does not throw an error	Does not throw an error	

3	Test that adding all candidates to the Party works as intended		Does not throw a compilation error;	Does not throw a compilation error;	
4	Test that getCandidates matches the test vector		Does not throw an error; green.getCandidates() == test;	Does not throw an error; green.getCandidates() == test;	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date:** **3/25/2024**

Test Case ID#: Party_getCandidate_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the getCandidate() function correctly returns the correct candidate given the index.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getCandidate()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate class works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party/Candidate objects for testing	Party green(“Green”) Candidate ashwin; Candidate ashwin2; Candidate ashwin3; Candidate ashwin4; Candidate ashwin5;	Does not throw an error	Does not throw an error	
3	Test that adding all candidates to the Party works as intended		Does not throw a compilation error;	Does not throw a compilation error;	
4	Test that getCandidate() matches returns the correct index		Does not throw an error; ashwin-ashwin5 returns indexes 0-4 respectively.	Does not throw an error; ashwin-ashwin5 returns indexes 0-4 respectively.	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_getCandidateCount_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the getCandidateCount() function correctly returns the amount of candidates in the vector.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getCandidateCount()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate class works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**

2	Set up necessary Party/Candidate objects for testing	Party green("Green")
		Candidate ashwin;
		Candidate ashwin2;
		Candidate ashwin3;
		Candidate ashwin4;
		Candidate ashwin5;
			Does not throw an error	Does not throw an error	

3	Test that adding all candidates to the Party works as intended		Does not throw a compilation error;	Does not throw a compilation error;	
4	Test that getCandidateCount() matches returns the correct number		Does not throw an error; getCandidateCount() returns 5;	Does not throw an error; getCandidateCount() returns 5;	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_toString_Test1 **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the toString() function correctly formats the string when there are candidates in the vector.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::toString()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Candidate class works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**

2	Set up necessary Party/Candidate objects for testing	Party green(“Green”)
		Party green2(“Green 2”)
		Candidate ashwin;
		Candidate ashwin2;
		Candidate ashwin3;
		Candidate ashwin4;
		Candidate ashwin5;
			Does not throw an error	Does not throw an error	

3	Test that adding all candidates to the Party works as intended		Does not throw a compilation error;	Does not throw a compilation error;	
4	Test that toString() outputs the intended string with the party name and candidate names		Does not throw an error; toString returns “Green - [Ashwin, …, Ashwin5]	Does not throw an error; toString returns “Green - [Ashwin, …, Ashwin5]	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_toString_Test2 **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the toString() function correctly formats the string when there are no candidates in the vector.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::toString()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Candidate class works as intended

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1					
2	Set up necessary Party/Candidate objects for testing	Party green2(“Green 2”)	Does not throw an error	Does not throw an error	
3	Test that toString() outputs the intended string with the party name and empty brackets		Does not throw an error; toString returns “Green 2 - []”	Does not throw an error; toString returns “Green 2 - []”	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_getSeats_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the getSeats() function correctly gets the number of seats allotted to a party

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getSeats()

Automated: yes X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: setSeats() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Party green("Green")	Does not throw an error	Does not throw an error	
3	Check that getSeats() returns 0		Does not throw an error; green.getSeats() returns 0	Does not throw an error; green.getSeats() returns 0	
4	Set the number of seats to 15 and see if getSeats returns 15		Does not throw an error; green.getSeats() returns 15	Does not throw an error; green.getSeats() returns 15	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_setSeats_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the setSeats() function correctly sets the number of seats allotted to a party

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::setSeats()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: getSeats() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Party green("Green")	Does not throw an error	Does not throw an error	
3	Set the number of seats to 2000 and check that it's 2000 using getSeats()		Does not throw an error; green.getSeats() returns 2000	Does not throw an error; green.getSeats() returns 2000	
4	Set the number of seats to 0 and check that it's 0 using getSeats()		Does not throw an error; green.getSeats() returns 0	Does not throw an error; green.getSeats() returns 0	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __

Test Date: **3/25/2024**

Test Case ID#: Party_incSeats_Test

Name(s) of Testers: Ashwin Wariar

Test Description:
Test for class Party to test that the incSeats() function correctly increments the number of seats by 1.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::incSeats()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: getSeats() works as intended

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Party/Candidate objects for testing	Party green("Green")	Does not throw an error	Does not throw an error	
3	Set the number of seats to 5 and use incSeats() to increase it by 1		Does not throw an error; green.getSeats() returns 6	Does not throw an error; green.getSeats() returns 6	
4	Use incSeats() 4 times to increase it to 10		Does not throw an error; green.getSeats() returns 10	Does not throw an error; green.getSeats() returns 10	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_getRemainder_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the getRemainder() function correctly gets the remainder.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getRemainder()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: setRemainder() works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party/Candidate objects for testing	Party green(“Green”)	Does not throw an error	Does not throw an error	
3	Get the remainder after the party is instantiated		Does not throw an error; green.getRemainder() returns 0;	Does not throw an error; green.getRemainder() returns 0;	
4	Use setRemainder(4) to set remainder to 4		Does not throw an error; green.getRemainder() returns 4;	Does not throw an error; green.getRemainder() returns 4;	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_getName_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the getName() function correctly gets the name of the party.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::getName()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Constructor works as intended

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Set up necessary Party/Candidate objects for testing	Party green("Green") Party democratic("Democratic") Party republican("Republican")	Does not throw an error	Does not throw an error	
2	Use getName() to get the name of the parties		Does not throw an error; green.getName() returns "Green" republican.getName() returns "Republican" democratic.getName() returns "Democratic"	Does not throw an error; green.getName() returns "Green" republican.getName() returns "Republican" democratic.getName() returns "Democratic"	
3					
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_setRemainder_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the setRemainder() function correctly sets the remainder.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::setRemainder()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: getRemainder() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Party green("Green")	Does not throw an error	Does not throw an error	
3	Set the remainder to 6 after the party is instantiated		Does not throw an error; green.getRemainder() returns 6;	Does not throw an error; green.getRemainder() returns 6;	
4	Use setRemainder(0) to set remainder to 0		Does not throw an error; green.getRemainder() returns 0;	Does not throw an error; green.getRemainder() returns 0;	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Party_setName_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Party to test that the setName() function correctly sets the name of the party.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/party_unittest; Party(std::string name) and Party::setName()

Automated: yes_X__ no ___
Results: Pass __X___ Fail_________

Preconditions for Test: getName() works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party/Candidate objects for testing	Party democratic("Green")	Does not throw an error	Does not throw an error	
3	Set the name to "Democratic"		Does not give an error; democratic.getName() returns "Democratic"	Does not give an error; democratic.getName() returns "Democratic"	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_getVotes_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the getVotes() function correctly gets the votes for a candidate/party.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Electable::getVotes()

Automated: yes X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate and Party work as intended, Electable::addVotes() works as intended

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Party/Candidate objects for testing	Candidate c1("Ashwin") Party p1("Green")	Does not throw an error	Does not throw an error	
3	Make sure the number of votes are instantiated to zero		Does not give an error; c1.getvotes() and p1.getVotes() return 0	Does not give an error; c1.getvotes() and p1.getVotes() return 0	
4	Add votes to both objects and see if getVotes can get the votes		Does not give an error; c1.getvotes() returns 1000 and p1.getVotes() return 2342	Does not give an error; c1.getvotes() returns 1000 and p1.getVotes() return 2342	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_incVotes_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the incVotes() function correctly increments the votes for a candidate/party by 1.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Electable::incVotes()

Automated: yes X___ no ____

Results: Pass __X___ Fail________

Preconditions for Test: Candidate and Party work as intended, Electable::getVotes() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Candidate c1("Ashwin") Party p1("Green")	Does not throw an error	Does not throw an error	
3	Increment c1 by 1		Does not give an error; c1.getvotes() returns 1	Does not give an error; c1.getvotes() returns 1	
4	Increment c1 by 3 and p1 by 2		Does not give an error; c1.getvotes() returns 4 and p1.getVotes() returns 2	Does not give an error; c1.getvotes() returns 4 and p1.getVotes() returns 2	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_addVotes_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the addVotes() function correctly adds a certain amount of votes for a candidate/party.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Electable::addVotes()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate and Party work as intended

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1					
2	Set up necessary Party/Candidate objects for testing	Candidate c1(“Ashwin”) Party p1(“Green”)	Does not throw an error	Does not throw an error	
3	Add 555 votes to both p1 and c1		Does not give an error; c1.getvotes() and p1.getVotes() return 555	Does not give an error; c1.getvotes() and p1.getVotes() return 555	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_getFirstAllocation_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the getFirstAllocation() function correctly gets the seats for first allocation.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest;
Electable::getFirstAllocation()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party works as intended; setFirstAllocation() works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party/Candidate objects for testing	Party p1("Green")	Does not throw an error	Does not throw an error	
3	Get first allocation after p1 is instantiated		Does not give an error; p1.getFirstAllocation() returns 0	Does not give an error; p1.getFirstAllocation() returns 0	
4	Change first allocation to 12		Does not give an error; p1.getFirstAllocation() returns 12	Does not give an error; p1.getFirstAllocation() returns 12	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_setFirstAllocation_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the setFirstAllocation() function correctly sets the seats for first allocation.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Electable::setFirstAllocation()

Automated: yes X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party works as intended; getFirstAllocation() works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party/Candidate objects for testing	Party p1("Green")	Does not throw an error	Does not throw an error	
3	Set first allocation seats to 6 and check with getFirstAllocation()		Does not give an error; p1.getFirstAllocation() returns 6	Does not give an error; p1.getFirstAllocation() returns 6	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_getSecondAllocation_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the getSecondAllocation() function correctly gets the seats for second allocation.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest;
Electable::getSecondAllocation()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Party works as intended; setSecondAllocation() works as intended

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Party/Candidate objects for testing	Party p1(“Green”)	Does not throw an error	Does not throw an error	
3	Get second allocation after p1 is instantiated		Does not give an error; p1.getFirstAllocation() returns 0	Does not give an error; p1.getFirstAllocation() returns 0	
4	Change second allocation to 3		Does not give an error; p1.getSecondAllocation() returns 3	Does not give an error; p1.getSecondAllocation() returns 3	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_setSecondAllocation_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the setSecondAllocation() function correctly sets the seats for second allocation.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest;
Electable::setSecondAllocation()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party works as intended; getSecondAllocation() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Party p1("Green")	Does not throw an error	Does not throw an error	
3	Set first allocation after p1 is instantiated		Does not give an error; p1.getFirstAllocation() returns 3	Does not give an error; p1.getFirstAllocation() returns 3	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_incSecondAllocation_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the incSecondAllocation() function correctly increments the seats for second allocation by 1.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest;
Electable::incSecondAllocation()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Party works as intended; getSecondAllocation() works as intended

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Party/Candidate objects for testing	Party p1(“Green”)	Does not throw an error	Does not throw an error	
3	Increment the second allocation three times by calling the func three times		Does not give an error; p1.getSecondAllocation() returns 3	Does not give an error; p1.getSecondAllocation() returns 3	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_setName_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the setName() function correctly sets name for the party/candidate.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Party::setName(), Candidate::setName()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party/Candidate works as intended;

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1					
2	Set up necessary Party/Candidate objects for testing	Party p1("Green") Candidate c1("Ashwin")	Does not throw an error	Does not throw an error	
3	Set name for both candidate and Party and check using getName()		Does not give an error; p1.getName() returns "Democratic" and c1.getName() returns "Rob"	Does not give an error; p1.getName() returns "Democratic" and c1.getName() returns "Rob"	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_getName_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the getName() function correctly gets the name of party/candidate.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Electable::getName()

Automated: yes X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate/Party works as intended;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Party p1(“Green”) Candidate c1(“Ashwin”)	Does not throw an error	Does not throw an error	
3	Call getName() on both objects		Does not give an error; p1.getName() returns “Green” and c1.getName() returns “Ashwin”	Does not give an error; p1.getName() returns “Green” and c1.getName() returns “Ashwin”	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Electable_toString_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for abstract class Electable to test that the toString() function correctly formats the party/candidate with its information

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/electable_unittest; Electable::toString()

Automated: yes X no ___

Results: Pass __X___ Fail________

Preconditions for Test: Candidate/Party works as intended; Party::addCandidate() and Candidate::setParty() work;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Party/Candidate objects for testing	Party p1("Green") Candidate c1("Ashwin")	Does not throw an error	Does not throw an error	
3	Set party to p1 for c1 and add candidate c1 to party p1		Does not throw an error	Does not throw an error	
4	Call toString on both objects		Does not give an error; c1.toString() returns "Ashwin - Green" and p1.toString() returns "Green - [Ashwin]"	Does not give an error; c1.toString() returns "Ashwin - Green" and p1.toString() returns "Green - [Ashwin]"	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Candidate_Constructor_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Candidate to test that the Constructor correctly instantiates with a given name for the candidate

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest;
Candidate::Candidate(std::string name)

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Candidate::getName() work;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Candidate objects for testing	Candidate c1("Ashwin")	Does not throw an error	Does not throw an error	
3	Check that instantiating the constructor doesn't break anything		Does not throw an error	Does not throw an error	
4	Call getName() to make sure name is correct		Does not give an error; c1.getName() returns "Ashwin"	Does not give an error; c1.getName() returns "Ashwin"	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Candidate_setParty_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Candidate to test that setParty() works correctly to set the party of a candidate

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest; Candidate::setParty()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party works as intended; Candidate::getParty() works;

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					
2	Set up necessary Candidate/Party objects for testing	Candidate c1("Ashwin") Candidate c2("Ashwin Two") Candidate c3("Ashwin Three") Party green("Green") Party democratic("Democratic") Party republican("Republican")	Does not throw an error	Does not throw an error	
3	Set c1 to Green, c2 to Democratic, c3 to Republican		Does not throw an error	Does not throw an error	
4	Call getParty() to make sure party is correct for candidate		Does not give an error; c1.getParty() returns &green, c2.getParty() returns &democratic, c3.getParty() returns &republican	Does not give an error; c1.getParty() returns &green, c2.getParty() returns &democratic, c3.getParty() returns &republican	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Candidate_getParty_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Candidate to test that the getParty() correctly gets the party of a candidate

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest; Candidate::getParty()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party works as intended; Candidate::setParty() works;

Step # 1	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
2	Set up necessary Candidate/Party objects for testing	Candidate c1("Ashwin") Party green("Green")	Does not throw an error	Does not throw an error	
3	Set c1 to Green		Does not throw an error	Does not throw an error	
4	Call getParty() to make sure party is correct for candidate		Does not give an error; c1.getParty() returns &green,	Does not give an error; c1.getParty() returns &green	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Candidate_toString_Test1 **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Candidate to test that the toString() correctly outputs a candidate that has an associated party

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest; Candidate::toString()

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Party works as intended; Candidate::setParty() works;

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Candidate/Party objects for testing	Candidate c1("Ashwin") Party green("Green")	Does not throw an error	Does not throw an error	
3	Set c1 to Green		Does not throw an error	Does not throw an error	
4	Call toString() to make sure output is correct		Does not give an error; c1.toString() returns "Ashwin - Green"	Does not give an error; c1.toString() returns "Ashwin - Green"	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Candidate_toString_Test2 **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Candidate to test that the toString() correctly outputs a candidate that doesn't have an associated party

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest; Candidate::toString()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Party works as intended; Candidate::setParty() works;

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Candidate/Party objects for testing	Candidate c2("Ashwin Two")	Does not throw an error	Does not throw an error	
3	Call toString() to make sure output is correct		Does not throw an error; c2.toString() outputs "Ashwin Two - No party"	Does not throw an error; c2.toString() outputs "Ashwin Two - No party"	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __

Test Date: 3/25/2024

Test Case ID#: Candidate_getName_Test

Name(s) of Testers: Ashwin Wariar

Test Description:
Test for class Candidate to test that the getName() function correctly outputs a candidate's name

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest; Candidate::getName()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: None

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Candidate/Party objects for testing	Candidate c2("Ashwin Two")	Does not throw an error	Does not throw an error	
3	Call getName() to make sure output is correct		Does not throw an error; c2.getName() outputs "Ashwin Two"	Does not throw an error; c2.getName() outputs "Ashwin Two"	
4					

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System __ **Test Date: 3/25/2024**

Test Case ID#: Candidate_setName_Test **Name(s) of Testers:** Ashwin Wariar

Test Description:
Test for class Candidate to test that the setName() function correctly sets a candidate's name

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/candidate_unittest; Candidate::setName()

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Candidate::getName() works as intended;

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Set up necessary Candidate/Party objects for testing	Candidate c1("Ashwin")	Does not throw an error	Does not throw an error	
3	Call setName() to change the name to "Rob"		Does not throw an error;	Does not throw an error;	
4	Call getName() to check the name		Does not throw an error; c1.getName() outputs "Rob"	Does not throw an error; c1.getName() outputs "Rob"	

Post condition(s) for Test:
None

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: createWindowTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests that guiWindow can be constructed. The tester may also verify that the window has opened.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/guiwindow_unittest.cc
guiWindow::guiWindow();

Automated: yes X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: Some .csv file exists and can be selected. All dependencies installed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Gtk application	auto app	Is created	Is created	
2	Construct guiWindow	guiWindow::guiWindow winTest	Does not throw error	Does not throw error	

Post condition(s) for Test:
guiWindow can be constructed.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: getFNameFromWindowTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests that the guiWindow returns the selected filename of type .csv. The tester may also verify that the window opens and can select a .csv file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/guiwindow_unittest.cc
guiWindow::guiWindow(); guiWindow::getFilename();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: Some .csv file exists and can be selected. All dependencies installed. guiWindowCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Gtk application	auto app	Is created	Is created	
2	Construct guiWindow	guiWindow::guiWindow gui	Is created	Is created	
3	Get file name	guiWindow::getFilename()	Does not throw error	Does not throw error	
4	Check file name is not empty	std::string fname	Is not “”	Is not “”	
5	Check file extension	std::string fname	Is “.csv”	Is “.csv”	

Post condition(s) for Test:
getFilename returns selected filename.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLCreateTest **Name(s) of Testers:** Alex Johnson

Test Description:
Creates an Audit object for an OPL election.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);

Automated: yes_X__ no ____

Results: Pass __X___ Fail_________

Preconditions for Test: VotingSystem has been created and the election has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	No error	No error	

Post condition(s) for Test:
Audit can be constructed.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date:** **3/25/2024**

Test Case ID#: AuditOPLWriteElectionTypeTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing the election type.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeElectionType();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write election type	Audit::writeElectionType()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can write election type correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: 3/25/2024

Test Case ID#: AuditOPLWritePartyCountTest

Name(s) of Testers: Alex Johnson

Test Description:
Tests writing party count.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writePartyCount();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write party count	Audit::writePartyCount()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write party count.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWriteBallotCountTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing ballot count.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeBallotCount();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write ballot count	Audit::writeBallotCount()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write ballot count.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWriteSeatCountTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing seat count.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeSeatCount();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write seat count	Audit::writeSeatCount()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write seat count.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWritePartyTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing party name.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writePartyName(); Party::Party(std::string);

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditOPLCreateTest passed. Can create Party obejcts

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Construct Party object	Party::Party()	new Party is created	new Party is created	
3	Write party name	Audit::writePartyName()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write party name.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: **3/25/2024**

Test Case ID#: AuditOPLWriteAllPartiesTest

Name(s) of Testers: Alex Johnson

Test Description:

Tests writing all parties.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.

Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem); Audit::writeParty();
Audit::writeAllParties();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditOPLCreateTest passed. AuditOPLWritePartyTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write all parties	Audit::writeAllParties()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:

Can correctly write all parties.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWriteEquationTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing quota equation.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeEquation();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write equation	Audit::writeEquation()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write equation.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWriteTableTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing summary table.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem); Audit::writeTable();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write table	Audit::writeTable()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write summary table.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWriteWinnersTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing all winners.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeWinners();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write winners	Audit::writeWinners()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write winners.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditOPLWriteResultsTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing full results to file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditOPLTest.csv
Project1/testing/BigOPL.csv
Project1/src/test/audit_unittest/audit.html
Project1/src/test/audit_unittest/correctopltest.html
Project1/src/test/audit_unittest/correctbopltest.html
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
OPL::processElectables(); OPL::countVotes();
OPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeResults(); Audit::writeElectionType();
Audit::writePartyCount(); Audit::writeBallotCount();
Audit::writeSeatCount(); Audit::writeParty();
Audit::writeAllParties(); Audit::writeEquation();
Audit::writeTable(); Audit::writeWinners(); Audit::writeResults();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditOPLCreateTest passed. All prior Audit OPL tests passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	

2	Write results	Audit::writeResults()	No error	No error	
3	Open answer file	std::ifstream co; “correctoplaudit.html”	The file is opened into the filestream	The file is opened into the filestream	
4	Read in answer file	std::stringstream cob	The file is read into the string stream	The file is read into the string stream	
5	Open actual file	std::ifstream o; “audit.html”	The file is opened into the filestream	The file is opened into the filestream	
6	Read in actual file	std::stringstream ob	The file is read into the string stream	The file is read into the string stream	
7	Compare files	std::stringstream o; std::stringstream co	No error and is equal to test file	No error and is equal to test file	
8	Construct Audit object	Audit::Audit ba	Audit ba is created	Audit ba is created	
9	Write results	Audit::writeResults()	No error	No error	
10	Open big answer file	std::ifstream bco; “correctboplaudit.html”	The file is opened into the filestream	The file is opened into the filestream	
11	Read in big answer file	std::stringstream bcob	The file is read into the string stream	The file is read into the string stream	
12	Open actual big file	std::ifstream bo; “audit.html”	The file is opened into the filestream	The file is opened into the filestream	
13	Read in actual big file	std::stringstream bob	The file is read into the string stream	The file is read into the string stream	
14	Compare big files	std::stringstream bo; std::stringstream bco	No error and is equal to test file	No error and is equal to test file	

Post condition(s) for Test:

Can correctly write results to file. Creates audit.html.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: 3/25/2024

Test Case ID#: AuditCPLCreateTest

Name(s) of Testers: Alex Johnson

Test Description:
Creates an Audit object for an CPL election.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: VotingSystem has been created and the election has been processed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	No error	No error	

Post condition(s) for Test:
Audit can be constructed.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteElectionTypeTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing the election type.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeElectionType();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write election type	Audit::writeElectionType()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can write election type correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date:** **3/25/2024**

Test Case ID#: AuditCPLWritePartyCountTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing party count.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writePartyCount();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write party count	Audit::writePartyCount()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write party count.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteBallotCountTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing ballot count.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeBallotCount();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write ballot count	Audit::writeBallotCount()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write ballot count.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteSeatCountTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing seat count.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeSeatCount();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write seat count	Audit::writeSeatCount()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write seat count.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWritePartyTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing party name.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writePartyName(); Party::Party(std::string);

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditCPLCreateTest passed. Can create Party obejcts

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Construct Party object	Party::Party()	new Party is created	new Party is created	
3	Write party name	Audit::writePartyName()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write party name.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteAllPartiesTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing all parties.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem); Audit::writeParty();
Audit::writeAllParties();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditCPLCreateTest passed. AuditCPLWritePartyTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write all parties	Audit::writeAllParties()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write all parties.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteEquationTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing quota equation.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeEquation();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write equation	Audit::writeEquation()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write equation.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteTableTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing summary table.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem); Audit::writeTable();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write table	Audit::writeTable()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write summary table.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteWinnersTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing all winners.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeWinners();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: AuditCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	
2	Write winners	Audit::writeWinners()	No error and is equal to test string	No error and is equal to test string	

Post condition(s) for Test:
Can correctly write winners.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: AuditCPLWriteResultsTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests writing full results to file.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/audit_unittest/audit_unittest.cc
Project1/testing/AuditCPLTest.csv
Project1/testing/BigCPL.csv
Project1/src/test/audit_unittest/audit.html
Project1/src/test/audit_unittest/correctcpltest.html
Project1/src/test/audit_unittest/correctbcpltest.html
VotingSystemFactory::VotingSystemFactory();
VotingSystemFactory::newVotingSytem(std::string);
CPL::processElectables(); CPL::countVotes();
CPL::calculateResults();
Audit::Audit(VotingSystem::VotingSystem);
Audit::writeResults(); Audit::writeElectionType();
Audit::writePartyCount(); Audit::writeBallotCount();
Audit::writeSeatCount(); Audit::writeParty();
Audit::writeAllParties(); Audit::writeEquation();
Audit::writeTable(); Audit::writeWinners(); Audit::writeResults();

Automated: yes_X__ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: AuditCPLCreateTest passed. All prior Audit CPL tests passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Audit object	Audit::Audit a	Audit a is created	Audit a is created	

2	Write results	Audit::writeResults()	No error	No error	
3	Open answer file	std::ifstream co; “correctcplaudit.html”	The file is opened into the filestream	The file is opened into the filestream	
4	Read in answer file	std::stringstream cob	The file is read into the string stream	The file is read into the string stream	
5	Open actual file	std::ifstream o; “audit.html”	The file is opened into the filestream	The file is opened into the filestream	
6	Read in actual file	std::stringstream ob	The file is read into the string stream	The file is read into the string stream	
7	Compare files	std::stringstream o; std::stringstream co	No error and is equal to test file	No error and is equal to test file	
8	Construct Audit object	Audit::Audit ba	Audit ba is created	Audit ba is created	
9	Write results	Audit::writeResults()	No error	No error	
10	Open big answer file	std::ifstream bco; “correctbcplaudit.html”	The file is opened into the filestream	The file is opened into the filestream	
11	Read in big answer file	std::stringstream bcob	The file is read into the string stream	The file is read into the string stream	
12	Open actual big file	std::ifstream bo; “audit.html”	The file is opened into the filestream	The file is opened into the filestream	
13	Read in actual big file	std::stringstream bob	The file is read into the string stream	The file is read into the string stream	
14	Compare big files	std::stringstream bo; std::stringstream bco	No error and is equal to test file	No error and is equal to test file	

Post condition(s) for Test:

Can correctly write results to file. Creates audit.html.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: ElectionOPLCreateTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests constructing an Election object.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditOPLTest.csv
Election::Election(std::string);

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: A .csv file is available to be read. An Audit object can be used and created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	

Post condition(s) for Test:
Can correctly construct Election.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: ElectionOPLDoElectionTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests running and election.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditOPLTest.csv
Election::Election(std::string); Election::doElection();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: ElectionOPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	

Post condition(s) for Test:
Can correctly run Election.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: 3/25/2024

Test Case ID#: ElectionOPLDoAuditTest

Name(s) of Testers: Alex Johnson

Test Description:
Tests auditing an election.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditOPLTest.csv
Election::Election(std::string); Election::doElection();
Election()::doAudit();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: ElectionOPLDoElectionTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	
3	Audit the election	Election::doAudit()	No error	No error	

Post condition(s) for Test:
Can correctly audit Election.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: 3/25/2024

Test Case ID#: ElectionOPLDisplayTest

Name(s) of Testers: Alex Johnson

Test Description:
Tests displaying election results.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditOPLTest.csv
Election::Election(std::string); Election::doElection();
Election()::display();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: ElectionOPLDoElectionTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	
3	Display the election	Election::display()	No error	No error	
4	Check captured stdout for correctness	std::string output	output matches testing string	output matches testing string	

Post condition(s) for Test:
Can correctly display Election results.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: ElectionCPLCreateTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests constructing an Election object.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditCPLTest.csv
Election::Election(std::string);

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: A .csv file is available to be read. An Audit object can be used and created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	

Post condition(s) for Test:
Can correctly construct Election.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: ElectionCPLDoElectionTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests running and election.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditCPLTest.csv
Election::Election(std::string); Election::doElection();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: ElectionCPLCreateTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	

Post condition(s) for Test:
Can correctly run Election.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: ElectionCPLDoAuditTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests auditing an election.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditCPLTest.csv
Election::Election(std::string); Election::doElection();
Election()::doAudit();

Automated: yes_X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: ElectionCPLDoElectionTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	
3	Audit the election	Election::doAudit()	No error	No error	

Post condition(s) for Test:
Can correctly audit Election.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: ElectionCPLDisplayTest **Name(s) of Testers:** Alex Johnson

Test Description:
Tests displaying election results.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/election_unittest/election_unittest.cc
Project1/testing/AuditCPLTest.csv
Election::Election(std::string); Election::doElection();
Election()::display();

Automated: yes _X__ no ___

Results: Pass __X___ Fail________

Preconditions for Test: ElectionCPLDoElectionTest passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	
3	Display the election	Election::display()	No error	No error	
4	Check captured stdout for correctness	std::string output	output matches testing string	output matches testing string	

Post condition(s) for Test:
Can correctly display Election results.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: **3/24/2024**

Test Case ID#: Fileops_constructor_test

Name(s) of Testers: Leo Dong

Test Description:
The test objective is to test the constructor of Fileops

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/fileops_unittest/fileops_unittest.cc,
Project1/testing/cpl_ballot.csv,
Fileops::Fileops(std::string);
Fileops::Fileops();

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: No preconditions

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct default Fileops object	No test data	No error is thrown	No error is thrown	
2	Construct Fileops object with a non-existing file path	No test data	No error is thrown	No error is thrown	
3	Construct Fileops object with correct file path input	cpl_ballot.csv	No error is thrown	No error is thrown	
4					

Post condition(s) for Test:
Can correctly construct Fileops objects.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: 3/24/2024

Test Case ID#: Fileops_getFilename_test

Name(s) of Testers: Leo Dong

Test Description:
The test objective is to test the correctness of the getFilename() method of the Fileops class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/fileops_unittest/fileops_unittest.cc,
Project1/testing/Ballot.csv,
Project1/testing/textTestfile.txt,
Project1/testing/binTestFile.bin,
Project1/testing/errorFile,
Fileops::getFilename();

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: Fileops constructor functions properly by extracting the filename from a file path.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the default Fileops object with a default filename	No test data	“Unknown”	“Unknown”	
2	Test other Fileops objects with filenames extracted by the constructor	textTestfile.txt binTestFile.bin Ballot.csv errorFile	"textTestfile.txt" "binTestFile.bin" "Ballot.csv" "errorFile"	"textTestfile.txt" "binTestFile.bin" "Ballot.csv" "errorFile"	
3					
4					

Post condition(s) for Test: Fileops::getFilename() works as intended.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/24/2024**

Test Case ID#: Fileops_checkCSVFormat_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the helper function checkCSVFormatTest() of the Fileops class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/fileops_unittest/fileops_unittest.cc,
Project1/testing/Ballot.csv,
Project1/testing/cpl_ballot.csv,
Project1/testing/opl_ballot.csv,
Project1/testing/textTestfile.txt,
Project1/testing/binTestFile.bin,
Project1/testing/errorFile,
Fileops::checkCSVFormat();

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: Fileops constructor functions properly

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the false case with all the non-.csv files	textTestfile.csv binTestfile.bin	Throw error “Only .csv file is allowed”	Throw error “Only .csv file is allowed”	
2	Test the false case with incorrect filename format	errorFile	Throw error “Incorrect filename format”	Throw error “Incorrect filename format”	
3	Test the true cases with .csv files	Ballot.csv cpl_ballot.csv opl_ballot.csv	exit code = 0	exit code = 0	
4					

Post condition(s) for Test: Fileops::checkCSVFormat() works as intended.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/24/2024**

Test Case ID#: Fileops_write_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the write() method of the Fileops class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/fileops_unittest/fileops_unittest.cc,
Project1/src/tests/fileops_unittest/fileToWrite.txt,
A non-existing file,
Fileops::write();

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the if branch of the function	fileToWrite.txt	exit code = 0	exit code = 0	
2	Test the correctness of fileIO process	fileToWrite.txt	read line “Hello World” from fileToWrite.txt after writing to it	read line “Hello World” from fileToWrite.txt	
3	Test the else branch of the function	a non-existing file	Throw runtime error	Throw runtime error	
4					

Post condition(s) for Test: Fileops::write() works as intended.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: 3/24/2024

Test Case ID#: Fileops_parseFile_test

Name(s) of Testers: Leo Dong

Test Description:
The test objective is to test the correctness of the parseFile() method of the Fileops class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/fileops_unittest/fileops_unittest.cc,
Project1/src/tests/fileops_unittest/fileToWrite.txt,
Project1/src/tests/fileops_unittest/Ballot.csv,
A non-existing file,
Fileops::parseFile();

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the if branch of the function	fileToWrite.txt Ballot.csv	exit code = 0	exit code = 0	
2	Test the correctness of fileIO process	fileToWrite.txt Ballot.csv	read line “test\test\test” from fileToWrite.txt read line “Ballot” from Ballot.csv	read line “test\test\test” from fileToWrite.txt read line “Ballot” from Ballot.csv	
3	Test the else branch of the function	a non-existing file	Throw runtime error	Throw runtime error	
4					

Post condition(s) for Test: Fileops::parseError() works as intended.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___

Test Date: **3/24/2024**

Test Case ID#: Fileops_parseFile_test

Name(s) of Testers: Leo Dong

Test Description:
The test objective is to test the correctness of the parseFile() method of the Fileops class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/fileops_unittest/fileops_unittest.cc,
Project1/src/tests/fileops_unittest/fileToWrite.txt,
Project1/src/tests/fileops_unittest/Ballot.csv,
A non-existing file,
Fileops::parseFile();

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the if branch of the function	fileToWrite.txt Ballot.csv	exit code = 0	exit code = 0	
2	Test the correctness of fileIO process	fileToWrite.txt Ballot.csv	read line “test\test\test” from fileToWrite.txt read line “Ballot” from Ballot.csv	read line “test\test\test” from fileToWrite.txt read line “Ballot” from Ballot.csv	
3	Test the else branch of the function	a non-existing file	Throw runtime error	Throw runtime error	
4					

Post condition(s) for Test: Fileops::parseError() works as intended.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: RawData_constructor_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the RawData class constructors that are inherited from the Fileops class

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/rawData_unittest/rawData_unittest.cc,
Project1/testing/cpl_ballot.csv,
A non-existing file,
RawData::RawData(); RawData::RawData(std::string)

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct a default RawData object with no file path	None	No error is thrown	No error is thrown	
2	Construct a RawData object with given file path	cpl_ballot.csv	No error is thrown	No error is thorwn	
3					
4					

Post condition(s) for Test: RawData::RawData() and RawData::RawData(std::string filepath)works as intended.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date:** **3/25/2024**

Test Case ID#: RawData_write_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of write() method in the RawData class. It returns an error for all cases cause no raw data file should be overwritten at any scenario.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/rawData_unittest/rawData_unittest.cc,
Project1/testing/cpl_ballot.csv,
Project1/testing/opl_ballot.csv,
A non-existing file,
RawData::write(std::string writable)

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the function with a default RawData object with no file path	None	A runtime error is thrown	A runtime error is thrown	
2	Test the function with RawData objects with valid file path	cpl_ballot.csv opl_ballot.csv textTestfiel.csv	A runtime error is thrown	A runtime error is thrown	
3					
4					

Post condition(s) for Test: RawData::write() overwrites the Fileops::write() method.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: RawData_getter_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of all getter methods in the RawData class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/rawData_unittest/rawData_unittest.cc,
Project1/testing/cpl_ballot.csv,
RawData::getElection(); RawData::getElectables();
RawData::getBallot(); RawData::getSeat();
RawData::getElectablesInfo(); RawData::getBallotInfo()

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: RawData constructors work properly

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the function with a default RawData object with no file path	None	return value of : “”, -1, -1, -1, “”, “”	return value of : “”, -1, -1, -1, “”, “”	
2	Test the function with RawData object with a valid file path	cpl_ballot.csv	return value of : “”, -1, -1, -1, “”, “”	return value of : “”, -1, -1, -1, “”, “”	
3					
4					

Post condition(s) for Test: All getter methods, RawData::getElection(); RawData::getElectables(); RawData::getBallot(); RawData::getSeat(); RawData::getElectablesInfo(); RawData::getBallotInfo(), work as intended

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: RawData_getFilename_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the getFilename() in the RawData class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/rawData_unittest/rawData_unittest.cc,
Project1/testing/cpl_ballot.csv,
Project1/testing/opl_ballot.csv,
RawData::getFilename()

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the function with a default RawData object with no file path	None	a return value of “Unknown”	a return value of “Unknown”	
2	Test the function with RawData object with a valid file path	cpl_ballot.csv opl_ballot.csv	a return value of “cpl_ballot.csv”	a return value of “cpl_ballot.csv”	
3					
4					

Post condition(s) for Test: getFilename() works as intended

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/25/2024**

Test Case ID#: RawData_parseFile_test **Name(s) of Testers:** Leo Dong

Test Description:

The test objective is to test the correctness of the parFile() in the RawData class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/rawData_unittest/rawData_unittest.cc,
Project1/testing/cpl_ballot.csv,
Project1/testing/opl_ballot.csv,
RawData::parseFile()

Automated: __X__ no ______

Results: Pass __X___ Fail________

Preconditions for Test: all getter methods work as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct RawData objects with a cpl ballot file path and a opl ballot file path	cpl_ballot.csv opl_ballot.csv	No error	No error	
2	Calls parseFile() on these two object	None	No error	No error	
3	Compare the value of cpl raw data with .csv file using all getter methods	cpl_ballot.csv	Type = “CPL” eletablesCount = 6 ballotCount =9 seatCount = 3 elecatblesInfo = "Democratic, Joe, Sally, Ahmed\n" "Republican, Allen, Nikki, Taihui\n" "New Wave, Sarah\n" "Reform, Xinyue, Nikita\n" "Green, Bethany\n" "Independent, Mike\n"; ballotInfo = "1,,,,,\n"	Type = “CPL” eletablesCount = 6 ballotCount =9 seatCount = 3 elecatblesInfo = "Democratic, Joe, Sally, Ahmed\n" "Republican, Allen, Nikki, Taihui\n" "New Wave, Sarah\n" "Reform, Xinyue, Nikita\n" "Green, Bethany\n" "Independent, Mike\n"; ballotInfo = "1,,,,,\n" "1,,,,,\n"	

			"1,,,,,\n" ",1,,,,\n" ",,,,1,\n" ",,,,,1\n" ",,,1,,\n" ",,,1,,\n" "1,,,,,\n" ",1,,,,\n";	",1,,,,\n" ",,,,1,\n" ",,,,,1\n" ",,,1,,\n" ",,,1,,\n" "1,,,,,\n" ",1,,,,\n";	
4	Compare the value of cpl raw data with .csv file using all getter methods	opl_ballot.csv	type = “OPL” eletablesCount = 6 ballotCount =9 seatCount = 2 elecatblesInfo = "Democrat, Pike\n" "Democrat, Lucy\n" "Democrat, Beiye\n" "Republican, Etta\n" "Republican, Alawa\n" "Independent1, Sasha\n"; ballotInfo = "1,,,,,\n" ",1,,,,\n" ",1,,,,\n" ",,,,1,\n" ",,,,,1\n" ",,,1,,\n" ",,,,1,\n" ",,,,1,\n" ",,,,,1\n";	type = “OPL” eletablesCount = 6 ballotCount =9 seatCount = 2 elecatblesInfo = "Democrat, Pike\n" "Democrat, Lucy\n" "Democrat, Beiye\n" "Republican, Etta\n" "Republican, Alawa\n" "Independent1, Sasha\n"; ballotInfo = "1,,,,,\n" ",1,,,,\n" ",1,,,,\n" ",,,,1,\n" ",,,,,1\n" ",,,1,,\n" ",,,,1,\n" ",,,,1,\n" ",,,,,1\n";	
5	Test a rawData object with an invalid ballot file path	errorFile	Throw runtime error	Throw runtime error	

Post condition(s) for Test: RawData::parseFile() works as intended

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_constructor_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the CPL class constructor.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc,
Project1/testing/cpl_ballot.csv,
CPL::CPL()

Automated: _____ no ___X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Call the constructor	None	No error is thrown	No error is thrown	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2					
3					
4					

Post condition(s) for Test: CPL::CPL() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_getParties_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the getParties() method in CPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc,
Project1/testing/cpl_ballot.csv,
CPL::getParties()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up CPL object with correct ballot file	cpl_ballot.csv	No error is thrown	No error is thrown	To compie the cpl_unittest.cc, please manually uncomment line 34 and 35 in cpl.h, and line 11 and line 88 for the purpose of testing protected variables.
2	Check the default *Party* value in the cpl object	None	the Party vector size = 0	the Party vector size = 0	
3	Create and set a dummy Party vector value to the cpl object	new Party("Democratic") new Party("Republican")	the Party vector size = 2 Party[0]->getName = “Democratic"; Party[1]->getName = “Republican"	the Party vector size = 2 Party[0]->getName = “Democratic"; Party[1]->getName = “Republican"	
4	Remove a party from the Party vector and confirm again	None	the Party vector size = 1 Party[0]->getName = “Democratic";	the Party vector size = 1 Party[0]->getName = “Democratic";	

Post condition(s) for Test: CPL::getParties() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_processElectables_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the processElectables() method in CPL class.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL::processElectables()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended; getParties() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up CPL object with correct ballot file	cpl_ballot.csv	No error is thrown PartyCount = 0 CandidateCount = 0 Party vector size = 0	No error is thrown PartyCount = 0 CandidateCount = 0 Party vector size = 0	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	Call the processElectables()	None	No error is thrown	No error is thrown	

3	Compare class fields with actual ballot info	cpl_ballot.csv	Party vector = <Party("Democratic");
			Party("Republican");
			Party("New Wave");
			Party("Reform");
			Party("Green");
			Party("Independent")>
				Party vector =<Party("Democratic");
				Party("Republican");
				Party("New Wave");
				Party("Reform");
				Party("Green");
				Party("Independent")>	

4	Compare the logic of sorting candidates vector	cpl_ballot.csv	All candidates and associated parties are matching	All candidates and associated parties are matching	
5	Check party count value and candidate count to ensure class fields are being updated	None	partyCount = 6 candidateCount = 11	partyCount = 6 candidateCount = 11	

Post condition(s) for Test: CPL::processElectables() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_countVotes_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the countVotes() method in CPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL::countVotes()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up CPL object with correct ballot file and check default vote count value	cpl_ballot.csv	totalVotes = 0	totalVotes = 0	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	call getVotes() without processing the electables	None	No error is thrown totalVotes = 0	No error is thrown totalVotes = 0	
3	call getVotes() after processing the electables	cpl_ballot.csv	No error is thrown totalVotes = 9	No error is thrown totalVotes = 9	

4	Check electables field with actual data	cpl_ballot.csv	Votes and parties are map as:
			["Democratic"] = 3;
			["Republican"] = 2;
			["New Wave"] = 0;
			["Reform"] = 2;
			["Green"] = 1;
			["Independent"] = 1;
				Votes and parties are map as:
				["Democratic"] = 3;
				["Republican"] = 2;
				["New Wave"] = 0;
				["Reform"] = 2;
				["Green"] = 1;
				["Independent"] = 1;	

Post condition(s) for Test: CPL::countVotes() functions properly

Project Name: Project 1: Voting System Team# 13

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_calculateResultsSort_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the sorting logic in the calculateResults() method in CPL class.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL:: calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and match the electables vector with actual data	cpl_ballot.csv	Electables toString value = "Democratic - [Joe, Sally, Ahmed]\n" "Republican - [Allen, Nikki, Taihui]\n" "New Wave - [Sarah]\n" "Reform - [Xinyue, Nikita]\n" "Green - [Bethany]\n" "Independent - [Mike]\n";	Elecatbles toString value = "Democratic - [Joe, Sally, Ahmed]\n" "Republican - [Allen, Nikki, Taihui]\n" "New Wave - [Sarah]\n" "Reform - [Xinyue, Nikita]\n" "Green - [Bethany]\n" "Independent - [Mike]\n";	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the soring logic given the ballot file	cpl_ballot.csv	Electables toString value = "Democratic - [Joe, Sally, Ahmed]\n" "Republican - [Allen, Nikki, Taihui]\n" "Reform - [Xinyue, Nikita]\n" "Green - [Bethany]\n" "Independent - [Mike]\n" "New Wave - [Sarah]\n";	Electables toString value = "Democratic - [Joe, Sally, Ahmed]\n" "Republican - [Allen, Nikki, Taihui]\n" "Reform - [Xinyue, Nikita]\n" "Green - [Bethany]\n" "Independent - [Mike]\n" "New Wave - [Sarah]\n";	

4					

Post condition(s) for Test: the sorting logic in CPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_calculateResultsFirstAllocation_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the firstAllocation logic in the calculateResults() method in CPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL::calculateResults()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and check the default value before calling calculateResutls	cpl_ballot.csv	all firstAllocation value for the parties equals 0	all firstAllocation value for the parties equals 0	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the calculationlogic given the ballot file	cpl_ballot.csv	only the democratic gets a firstAlloction vote	only the democratic gets a firstAlloction vote	
4					

Post condition(s) for Test: the firstAllocation logic in CPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_calculateResultsSecondtAllocation_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the secondAllocation logic in the calculateResults() method in CPL class.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL::calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and check the default value before calling calculateResutls	cpl_ballot.csv	all secondAllocation value for the parties equals 0	all secondAllocation value for the parties equals 0	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the calculationlogic given the ballot file	cpl_ballot.csv	only the Republican and Reform get a secondAllocation vote	only the democratic gets a secondAllocation vote	
4					

Post condition(s) for Test: the secondAllocation logic in CPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_calculateResultsTotalSeats_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the secondAllocation logic in the calculateResults() method in CPL class.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL::calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and check the default value before calling calculateResutls	cpl_ballot.csv	all totalSeats value for the parties equals 0	all totalSeats value for the parties equals 0	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the calculationlogic given the ballot file	cpl_ballot.csv	Democratic, Republican, and Reform have totatSeats of 1	Democratic, Republican, and Reform have totatSeats of 1	
4					

Post condition(s) for Test: the TotalSeats logic in CPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: CPL_calculateResultsGetWinners_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the getWinners() method in CPL class.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/cpl_unittest/cpl_unittest.cc, Project1/testing/cpl_ballot.csv, CPL::getWinners()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended. all other class methods are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes(), and calculateResults to process rawdata	cpl_ballot.csv	no error is thrown	no error is thrown	CPL unit test is not an automated test. To compile the cpl_unittest.cc source file, please manually uncomment lines 35 and 37 in cpl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables..
2	call getWinners()	None	No error is thrown	No error is thrown	
3	Check the winner given the ballot file	cpl_ballot.csv	Democratic - Joe, Republican - Allen, and Reform - Xinyue won a seat	Democratic - Joe, Republican - Allen, and Reform - Xinyue won a seat	
4					

Post condition(s) for Test: the TotalSeats logic in CPL::getWinners() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_constructor_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the OPL class constructor.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc,
Project1/testing/opl_ballot.csv,
OPL::OPL()

Automated: __X___ no ______

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call the constructor	None	No error is thrown	No error is thrown	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2					
3					
4					

Post condition(s) for Test: OPL::OPL() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_getParties_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the getParties() method in OPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc,
Project1/testing/opl_ballot.csv,
OPL::getParties()

Automated: __X___ no _____

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up OPL object with correct ballot file	OPL_ballot.csv	No error is thrown	No error is thrown	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	Check the default *Party* value in the OPL object	None	the Party vector size = 0	the Party vector size = 0	
3	Create and set a dummy Party vector value to the OPL object	new Party("Democratic") new Party("Republican")	the Party vector size = 3 Party[0]->getName = “Democratic"; Party[1]->getName = “Republican" Party[2]->getName = “Independent"	tthe Party vector size = 3 Party[0]->getName = “Democratic"; Party[1]->getName = “Republican" Party[2]->getName = “Independent"	
4					

Post condition(s) for Test: OPL::getParties() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date:** 3/26/2024

Test Case ID#: OPL_processElectables_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the processElectables() method in OPL class.

Automated: __X___ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::processElectables()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended; getParties() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up OPL object with correct ballot file	OPL_ballot.csv	No error is thrown PartyCount = 0 CandidateCount = 0 Party vector size = 0	No error is thrown PartyCount = 0 CandidateCount = 0 Party vector size = 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	Call the processElectables()	None	No error is thrown	No error is thrown	
3	Compare class fields with actual ballot info	OPL_ballot.csv	Party vector = <Party("Democratic"); Party("Republican"); Party("Independent")>	Party vector =<Party("Democratic"); Party("Republican"); Party("Independent")>	
4	Compare the logic of sorting candidates vector	OPL_ballot.csv	All candidates and associated parties are matching	All candidates and associated parties are matching	
5	Check party count value and candidate count to ensure class fields are being updated	None	partyCount = 3 candidateCount = 6	partyCount = 3 candidateCount = 6	

Post condition(s) for Test: OPL::processElectables() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_countVotes_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the countVotes() method in OPL class.

Automated: __X___ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::countVotes()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up OPL object with correct ballot file and check default vote count value	OPL_ballot.csv	totalVotes = 0	totalVotes = 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call getVotes() without processing the electables	None	No error is thrown totalVotes = 0	No error is thrown totalVotes = 0	
3	call getVotes() after processing the electables	OPL_ballot.csv	No error is thrown totalVotes = 9	No error is thrown totalVotes = 9	

4	Check electables field with actual data	OPL_ballot.csv	Votes and parties are map as:
			["Pike"] = 3;
			["Lucy"] = 2;
			["Beiye"] = 0;
			["Etta"] = 2;
			["Alawa"] = 1;
			["Sasha"] = 1;
				Votes and parties are map as: ["Pike"] = 3;
				["Lucy"] = 2;
				["Beiye"] = 0;
				["Etta"] = 2;
				["Alawa"] = 1;
				["Sasha"] = 1;	

Post condition(s) for Test: OPL::countVotes() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_calculateResultsSort_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the sorting logic in the calculateResults() method in OPL class.

Automated: ___X__ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL:: calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and match the electables vector with actual data	OPL_ballot.csv	Electables toString value = "Democrat - [Pike, Lucy, Beiye]\n" "Republican - [Etta, Alawa]\n" "Independent - [Sasha]\n";	Elecatbles toString value = "Democrat - [Pike, Lucy, Beiye]\n" "Republican - [Etta, Alawa]\n" "Independent - [Sasha]\n";	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the soring logic given the ballot file	OPL_ballot.csv	Electables toString value = ""Democrat - [Lucy, Pike, Beiye]\n" "Republican - [Alawa, Etta]\n" "Independent - [Sasha]\n";	Electables toString value = "Democrat - [Lucy, Pike, Beiye]\n" "Republican - [Alawa, Etta]\n" "Independent - [Sasha]\n";	
4					

Post condition(s) for Test: the sorting logic in OPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_calculateResultsFirstAllocation_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the firstAllocation logic in the calculateResults() method in OPL class.

Automated: __X___ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/cpl_ballot.csv, OPL::calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and check the default value before calling calculateResutls	OPL_ballot.csv	all firstAllocation value for the parties equals 0	all firstAllocation value for the parties equals 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the calculationlogic given the ballot file	OPL_ballot.csv	only republican gets a firstAlloction vote	only republican gets a firstAlloction vote	
4					

Post condition(s) for Test: the firstAllocation logic in OPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_calculateResultsSecondtAllocation_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the secondAllocation logic in the calculateResults() method in OPL class.

Automated: __X___ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and check the default value before calling calculateResutls	OPL_ballot.csv	all secondAllocation value for the parties equals 0	all secondAllocation value for the parties equals 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the calculationlogic given the ballot file	OPL_ballot.csv	only the democrat get a secondAllocation vote	only the democratic gets a secondAllocation vote	
4					

Post condition(s) for Test: the secondAllocation logic in OPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date:** **3/26/2024**

Test Case ID#: OPL_calculateResultsTotalSeats_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the secondAllocation logic in the calculateResults() method in OPL class.

Automated: __X___ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::calculateResults()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes() to process rawdata and check the default value before calling calculateResutls	OPL_ballot.csv	all totalSeats value for the parties equals 0	all totalSeats value for the parties equals 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the calculationlogic given the ballot file	OPL_ballot.csv	Democratic, Republican have totatSeats of 1	Democratic, Republican have totatSeats of 1	
4					

Post condition(s) for Test: the TotalSeats logic in OPL::calculateResults() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/26/2024**

Test Case ID#: OPL_calculateResultsGetWinners_test **Name(s) of Testers:** Leo Dong

Test Description:
The test objective is to test the correctness of the getWinners() method in OPL class.

Automated: __X___ no _____

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::getWinners()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended. all other class methods are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and countVotes(), and calculateResults to process rawdata	OPL_ballot.csv	no error is thrown	no error is thrown	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call getWinners()	None	No error is thrown	No error is thrown	
3	Check the winner given the ballot file	OPL_ballot.csv	Democratic - Lucy, Republican - Alawa won a seat	Democratic - Lucy, Republican - Alawa won a seat	
4					

Post condition(s) for Test: the TotalSeats logic in OPL::getWinners() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit ___ System _X_ **Test Date: 3/26/2024**

Test Case ID#: OPL **Name(s) of Testers:** Alex Johnson

Test Description:
Tests the system with an OPL election

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/system_test/system_test.cc
Project1/testing/BigOPLTest.csv
Project1/src/test/system_test/audit.html
Project1/src/test/system_test/correctboplaudit.html
Election::Election(std::string); Election::doElection();
Election()::display();

Automated: ___X___ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: All unit tests passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	
3	Display the election	Election::display()	No error	No error	
4	Open answer file	std::ifstream co; “correctboplaudit.html”	The file is opened into the filestream	The file is opened into the filestream	
5	Read in answer file	std::stringstream cob	The file is read into the string stream	The file is read into the string stream	
6	Open actual file	std::ifstream o; “audit.html”	The file is opened into the filestream	The file is opened into the filestream	
7	Read in actual file	std::stringstream ob	The file is read into the string stream	The file is read into the string stream	
8	Compare files	std::stringstream o; std::stringstream co	No error and is equal to test file	No error and is equal to test file	

9	Check captured stdout for correctness	std::string output	output matches testing string	output matches testing string	

Post condition(s) for Test:

System functions as specified. Audit file created.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit ___ System _X_ **Test Date: 3/26/2024**

Test Case ID#: CPL **Name(s) of Testers:** Alex Johnson

Test Description:
Tests the system with an CPL election

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/system_test/system_test.cc
Project1/testing/BigCPLTest.csv
Project1/src/test/system_test/audit.html
Project1/src/test/system_test/correctbcplaudit.html
Election::Election(std::string); Election::doElection();
Election()::display();

Automated: ___X___ no ___

Results: Pass __X___ Fail_________

Preconditions for Test: All unit tests passed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Construct Election object	Election::Election a	Election a is created with no error	Election a is created with no error	
2	Run the election	Election::doElection()	No error	No error	
3	Display the election	Election::display()	No error	No error	
4	Open answer file	std::ifstream co; “correctbcplaudit.html”	The file is opened into the filestream	The file is opened into the filestream	
5	Read in answer file	std::stringstream cob	The file is read into the string stream	The file is read into the string stream	
6	Open actual file	std::ifstream o; “audit.html”	The file is opened into the filestream	The file is opened into the filestream	
7	Read in actual file	std::stringstream ob	The file is read into the string stream	The file is read into the string stream	
8	Compare files	std::stringstream o; std::stringstream co	No error and is equal to test file	No error and is equal to test file	

9	Check captured stdout for correctness	std::string output	output matches testing string	output matches testing string	

Post condition(s) for Test:

System functions as specified. Audit file created.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: OPL_getParties_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test the correctness of the getParties() method in CPL class.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc,
Project1/testing/opl_ballot.csv,
OPL::getParties()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Set up OPL object with correct ballot file	opl_ballot.csv	No error is thrown	No error is thrown	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	Check the default *Party* value in the opl object	None	the Party vector size = 0	the Party vector size = 0	
3	Create and set a dummy Party vector value to the opl object	new Party("Democratic") new Party("Republican") newParty("Independent")	the ExpectedParties vector size = 3 ExpectedParties[0]->getName =“Democratic”; ExpectedParties[1]->getName = “Republican”; ExpectedParties[2]->getName =	the ExpectedParties vector size = 3 ExpectedParties[0]->getName =“Democratic”; ExpectedParties[1]->getName = “Republican”; ExpectedParties[2]->getName =	

			“Independent”	“Independent”	
4	Call processElectables() to load the parties into the opl object	None	No error is thrown	No error is thrown	@bug: Due to implementation choices, the opl object’s parties vector is only filled when processElectables() is called
5	Call getParties() and check that the retrieved values are accurate	ExpectedParties vector	the ActualParties vector size = 3 ActualParties[0]->getName =“Democratic”; ActualParties[1]->getName = “Republican”; ActualParties[2]->getName = “Independent”	the ActualParties vector size = 3 ActualParties[0]->getName =“Democratic”; ActualParties[1]->getName = “Republican”; ActualParties[2]->getName = “Independent”	

Post condition(s) for Test: OPL::getParties() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: OPL_processElectables_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test the correctness of the processElectables() method in OPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::processElectables()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended; getParties() works as intended

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up OPL object with correct ballot file	opl_ballot.csv	No error is thrown PartyCount = 0 CandidateCount = 0 Party vector size = 0	No error is thrown PartyCount = 0 CandidateCount = 0 Party vector size = 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	Set up dummy party vector containing party names that match the ballot party data	new Party("Democratic") new Party("Republican") newParty("Independent")	the ExpectedParties vector size = 3 ExpectedParties[0]->getName =“Democratic”; ExpectedParties[1]->getName = “Republican”; ExpectedParties[2]->getName = “Independent”	the ExpectedParties vector size = 3 ExpectedParties[0]->getName =“Democratic”; ExpectedParties[1]->getName = “Republican”; ExpectedParties[2]->getName = “Independent”	
3	Call the processElectables()	None	No error is thrown	No error is thrown	
4	Compare candidates and their associated parties	opl_ballot.csv	ActualCandidates[0]->toString() = “Pike - Democrat” ActualCandidates[1]->toString() = “Lucy - Democrat”	ActualCandidates[0]->toString() = “Pike - Democrat” ActualCandidates[1]->toString() = “Lucy - Democrat”	

			ActualCandidates[2]->toString() = "Beiye - Democrat" ActualCandidates[3]->toString() = "Etta - Republican" ActualCandidates[4]->toString() = "Alawa - Republican" ActualCandidates[5]->toString() = "Sasha - Independent"	ActualCandidates[2]->toString() = "Beiye - Democrat" ActualCandidates[3]->toString() = "Etta - Republican" ActualCandidates[4]->toString() = "Alawa - Republican" ActualCandidates[5]->toString() = "Sasha - Independent"	
5	Check party count value and candidate count to ensure class fields are being updated	None	partyCount = 3 candidateCount = 6	partyCount = 3 candidateCount = 6	

Post condition(s) for Test: OPL::processElectables() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: OPL_countVotes_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test the correctness of the countVotes() method in OPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL::countVotes()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up CPL object with correct ballot file and check default vote count value	cpl_ballot.csv	totalVotes = 0	totalVotes = 0	OPL unit test is not an automated test. To compile the opl_unittest.cc source file, please manually uncomment lines 34 and 36 in opl.h and lines 12 and 89 in votingsystem.h for the purpose of testing protected class variables.
2	call getVotes() without processing the electables	None	No error is thrown totalVotes = 0	No error is thrown totalVotes = 0	
3	call getVotes() after processing the electables	opl_ballot.csv	No error is thrown totalVotes = 9	No error is thrown totalVotes = 9	

4	Check electables field with actual data	opl_ballot.csv	Votes and candidates are map as:
			["Pike"] = 1;
			["Lucy"] = 2;
			["Beiye"] = 0;
			["Etta"] = 1;
			["Alawa"] = 3;
			["Sasha"] = 2;
				Votes and candidates are map as:
				["Pike"] = 1;
				["Lucy"] = 2;
				["Beiye"] = 0;
				["Etta"] = 1;
				["Alawa"] = 3;
				["Sasha"] = 2;	

Post condition(s) for Test: OPL::countVotes() functions properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: OPL_countVotesSort_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test the correctness of the countVotes() method in OPL class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/opl_unittest/opl_unittest.cc, Project1/testing/opl_ballot.csv, OPL:: countVotes()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, candidate, electable, party, fileops, rawdata, votingsystem classes, are functioning as intended.processElectables() and countVotes() are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElecatables() and getParties() to process rawdata and match the electables vector with actual data	opl_ballot.csv	Electables toString value = "Democrat - [Pike, Lucy, Beiye]\n" "Republican - [Etta, Alawa]\n" "Independent - [Sasha]\n";	Electables toString value = "Democrat - [Pike, Lucy, Beiye]\n" "Republican - [Etta, Alawa]\n" "Independent - [Sasha]\n";	For compiling the opl_unittest.cc, please manually uncomment line 34 and 35 in opl.h, and line 11 and line 88 for the purpose of testing protected variables.
2	call calculateResults()	None	No error is thrown	No error is thrown	
3	Check the sorting logic given the ballot file	opl_ballot.csv	Electables toString value = "Democrat - [Lucy, Pike, Beiye]\n" "Republican - [Alawa, Etta]\n" "Independent - [Sasha]\n";	Electables toString value = "Democrat - [Lucy, Pike, Beiye]\n" "Republican - [Alawa, Etta]\n" "Independent - [Sasha]\n";	

Post condition(s) for Test: the sorting logic in OPL::countVotes() works properly

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetAndSetBallotData_test **Name(s) of Testers:** Janani Kannan

Test Description:

The test objective is to test the get and set BallotData methods in VotingSystem.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getBallotData(), VotingSystem::setBallotData()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call setBallotData()	exampleOPLData = RawData(“opl_ballot.csv”) exampleCPLData = RawData(“cpl_ballot.csv”)	No error is thrown	No error is thrown	
2	compare ballot fields for pre-created OPL and CPL objects and ballot info obtained by calling getBallotData()	exampleOPLData = RawData(“opl_ballot.csv”) exOPL = new OPL() exampleCPLData = RawData(“cpl_ballot.csv”) exCPL = new CPL()	All ballot fields of pre-created exampleOPLData match ballot fields of the RawData object obtained by calling getBallotData()	All ballot fields of pre-created exampleOPLData match ballot fields of the RawData object obtained by calling getBallotData()	

Post condition(s) for Test: VotingSystem::getBallotData() and **VotingSystem::setBallotData()** function correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetBallotCount_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test getBallotCount method in VotingSystem functions as expected.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getBallotCount()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call getBallotCount() on pre-defined exampleOPLData and exampleCPLData	exampleOPLData = RawData("opl_ballot.csv") exampleCPLData = RawData("cpl_ballot.csv")	Both vote counts are 9.	Both vote counts are 9.	

Post condition(s) for Test: VotingSystem::getBallotCount() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetSeatCount_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test getSeatCount method in VotingSystem functions as expected.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getSeatCount()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	call getSeatCount() on pre-defined exampleOPLData and exampleCPLData	exampleOPLData = RawData("opl_ballot.csv") exampleCPLData = RawData("cpl_ballot.csv")	OPL and CPL seat count values from the Voting System objects match the seat counts extracted from RawData objects directly defined using the csv files	OPL and CPL seat count values from the Voting System objects match the seat counts extracted from RawData objects directly defined using the csv files	

Post condition(s) for Test: VotingSystem::getSeatCount() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetPartyCount_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test getPartyCount method in VotingSystem functions as expected.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getPartyCount()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	call processElectables() so that the parties list is populated	exCPL = new CPL(); exCPL->setBallotData(exampleCPLData)	No error thrown	No error thrown	
2	call getPartyCount() on pre-defined exCPL object; this is an exclusive CPL oriented method, and cant be tested on the OPL object since OPL handles party vectors differently.	exampleCPLData = RawData("cpl_ballot.csv")	CPL party count value from the Voting System object (exCPL) matches the party count extracted from RawData object (exampleCPLData) directly defined using the csv files	CPL party count value from the Voting System object (exCPL) matches the party count extracted from RawData object (exampleCPLData) directly defined using the csv files	

Post condition(s) for Test: VotingSystem::getPartyCount() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetCandidateCount_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test getCandidateCount method in VotingSystem functions as expected.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getCandidateCount()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call processElectables() so that the parties list is populated	exOPL = new OPL(); exOPL->setBallotData (exampleOPLData)	No error thrown	No error thrown	
2	call getCandidateCount() on pre-defined exOPL object; this is an exclusive OPL oriented method, and cant be tested on the CPL object since CPL handles candidate vectors differently.	exampleCPLData = RawData("cpl_ballot.csv")	OPL party count value from the Voting System object (exOPL) matches the party count extracted from RawData object (exampleOPLData) directly defined using the csv files	OPL party count value from the Voting System object (exOPL) matches the party count extracted from RawData object (exampleOPLData) directly defined using the csv files	

Post condition(s) for Test: VotingSystem::getCandidateCount() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_AddElectable_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test that addElectable() correctly adds the given electable to the VotingSystem object's existing electables list.

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::addElectable()

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	check if initial electable vector sizes are 0	exCPL = new CPL(); exCPL->setBallotData(exampleCPLData); exOPL = new OPL(); exOPL->setBallotData (exampleOPLData)	Electable sizes are 0	Electable sizes are 0	
2	call addElectable() on exCPL and exOPL objects using dummy party and candidate values	Party* partyToAdd("Democrat") Candidate* candidateToAdd("Janani")	No error thrown	No error thrown	
3	check the return value and the electable Name through getName() to confirm whether the add was successful	exCPL exOPL	return values are 0 to indicate successful add; partyname and candidatename matches the original party and candidate objects' names	return values are 0 to indicate successful add; partyname and candidatename matches the original party and candidate objects' names	

Post condition(s) for Test: VotingSystem::addElectable() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetElectables_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test that getElectables() correctly obtains the VotingSystem object's existing electables list.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getElectables()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create dummy party and candidate test data and call addElectable() on exOPL and exCPL objects to add these to the respective electables lists	Party p1("Democrat"); Party p2("Republican"); Party p3("Independent") Candidate c1("Janani") Candidate c2("Rose") Candidate c3("Amelia")	no error is thrown when adding electables	no error is thrown when adding electables	
2	compare party and candidate name fields of the added electables with expected party and candidate names to see if electables were correctly added	exCPL exOPL	exCPL-> getElectables() should contain "Democrat", "Republican" and "Independent" only. exOPL-> getElectables() should contain "Janani", "Rose" and "Amelia" only	exCPL-> getElectables() should contain "Democrat", "Republican" and "Independent" only. exOPL-> getElectables() should contain "Janani", "Rose" and "Amelia" only	

Post condition(s) for Test: VotingSystem::getElectables() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_GetElectionType_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test that getElectionType() correctly obtains the VotingSystem object's election type (either OPL or CPL).

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::getElectionType()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	pre-defined exOPL and exCPL objects in the setup() section of this class test; call getElectionType on each object	exCPL = new CPL(); exCPL->setBallotData(exampleCPLData); exOPL = new OPL(); exOPL->setBallotData (exampleOPLData)	No error thrown	No error thrown	
2	compare electionType values obtained through the call to getElectionType() with expected electionType values for exOPL and exCPL	exCPL exOPL	a getElectionType() call on exCPL produces "CPL" and on OPL produces "OPL"	a getElectionType() call on exCPL produces "CPL" and on OPL produces "OPL"	

Post condition(s) for Test: VotingSystem::getElectionType() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystem_resolveTie_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test that resolveTie() randomly picks one Electable object given a list of electables whose votecounts are tied.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystem_unittest/votingsystem_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystem::resolveTie()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	set up dummy data values for Party and Candidate vectors, each containing parties and candidates whose vote counts are tied. two different lengths of vectors: size 2 and 3	twoElectables and threeElectables; first testing with Party* objects, then with Candidate* objects	one electable is produced after resolveTie is called on twoElectables and threeElectables for both Party and Candidate objects	one electable is produced after resolveTie is called on twoElectables and threeElectables for both Party and Candidate objects	

Post condition(s) for Test: VotingSystem::resolveTie() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystemFactory_Constructor_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test that VotingSystemFactory() constructor creates a new object as intended

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystemfactory_unittest/votingsystemfactory_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystemFactory::VotingSystemFactory()

Automated: _____ no __X___

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call the constructor	None	No error is thrown	No error is thrown	

Post condition(s) for Test: VotingSystemFactory::VotingSystemFactory() functions correctly.

Project Name: Project 1: Voting System **Team# 13**

Test Stage: Unit _X_ System ___ **Test Date: 3/27/2024**

Test Case ID#: VotingSystemFactory_NewVotingSystem_test **Name(s) of Testers:** Janani Kannan

Test Description:
The test objective is to test that newVotingSystem() creates a new VotingSystem abstraction of either a CPL or OPL object as intended

Automated: _____ no __X___

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Project1/src/tests/votingsystemfactory_unittest/votingsystemfactory_unittest.cc, Project1/testing/opl_ballot.csv, Project1/testing/cpl_ballot.csv, VotingSystemFactory::newVotingSystem();

Results: Pass __X___ Fail________

Preconditions for Test: All dependencies, fileops, electables, rawdata, party, candidate, opl, cpl, and votingsystem classes, are functioning as intended.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call newVotingSystem() on a VotingSystemFactory object	None	No error is thrown	No error is thrown	
2	compare electionType field of the new object to the expected electionType	exOPL and exCPL	exOPL electionType() is “OPL” and exCPL electionType() is “CPL”	exOPL electionType() is “OPL” and exCPL electionType() is “CPL”	

Post condition(s) for Test: VotingSystemFactory::newVotingSystem() functions correctly.